
CERN
openlab

ML model registry in the Cloud

Renato Cardoso, Sofia Vallecorsa

Work realized in collaboration with Oracle

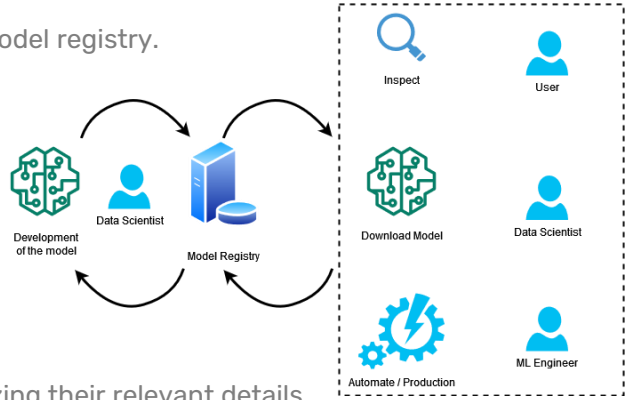
Model Registry

What is a model registry:

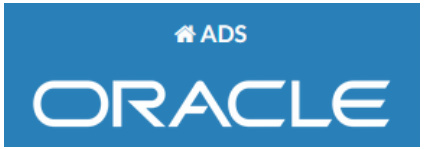
- A central repository, allowing to publish production-ready models, using a central UI to:
  - Search for models
  - View the status of models
  - View the necessary documentation
- A place to work together with other teams to manage the lifecycle of all models in the model registry.
- Models in the registry are ready to be tested, validated, and deployed to production.

Why use a model registry:

- Model registry enables faster deployment of your models and easy retrieval
- Model registry simplifies model lifecycle management
  - Track versions
  - Store metadata
  - Compare models
- Model registry enables production model governance by centralizing models and organizing their relevant details



Oracle Accelerated Data Science (ADS) SDK



What is the ADS SDK:

- Part of the Oracle Cloud Infrastructure (OCI) Data Science service.
- Speeds up common data science activities:
  - Manage the lifecycle of machine learning models
  - While providing resources for Data Flow, Object Storage, Autonomous Database

What can you do with ADS:

- Read datasets from Oracle Object Storage and other dataframes.
- Easily compute summary statistics and perform data profiling.
- **Tune models** using hyperparameter optimization.
- Generate detailed evaluation reports of your model.
- Save machine learning models (**Model Registry**).
- **Deploy** those models.
- **Train** machine learning models by itself or with Jobs.
- Manage the lifecycle of conda environments.
- **Distributed Training**

Model Registry in ADS framework

Model registry in Oracle is available through Oracle Accelerated Data Science SDK (ADS) named **Model Catalog**

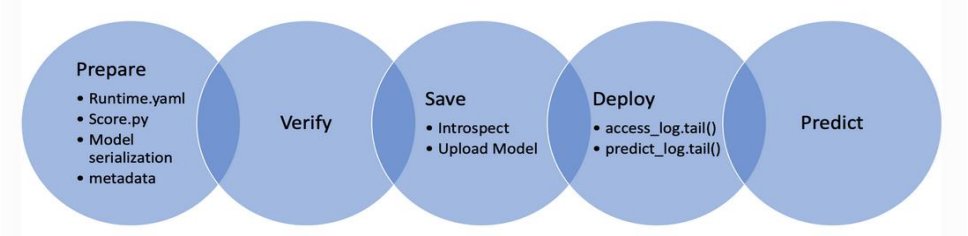
Oracle ADS model catalog let's the user:

- List, read, download, and load models from the catalog to their own notebook sessions
  - List all models on the model catalog
  - Download the model artifact from the catalog
  - Run the model on their own machine
  - Deploy the model artifact as a model deployment (being able to do inference on the model)
- Document the model use case and algorithm using taxonomy metadata.
- Add custom metadata that describes the model.
- Document the input data schema, and the returned inference schema.

Though the model catalog is mainly used to store ready for deployment models, it is still possible to save the training methods and later download them.

How to use the model Catalog

- Create a model serialization object.
- Use the .prepare() method to create the model artifacts
- Verify that everything is correctly setup
- Save the model in the model Catalog
- Deploy the model
- Predict the model results by parsing in data



Model Catalog Pipeline from ADS docs

How to use the model Catalog

- **Create a model serialization object.**
  - It wraps your model to assist in deploying it.
  - Different model classes:
    - PyTorch model -> the PyTorchModel class, TensorFlow model ->TensorFlowModel class, etc…
    - Not feasible to have a model serialization class for all model types -> use GenericModel
- **Use the .prepare() method to create the model artifacts.**
  - The score.py file is created to your model class.
    - Customize to your use case, i.e prediction method
  - Use it to store metadata about the model, code used to create the model, input and output schema, etc…
- **Verify that everything is correctly setup**
  - Call the .verify() method to confirm that the load_model() and predict() functions in this file are working.
    - This speeds up your debugging as you do not need to deploy a model to test it.
- **Save the model in the model Catalog**
  - The .save() method is then used to store the model in the model catalog.
- **Deploy the model**
  - Creates everything needed to perform inference on the model, using the deploy method.
- **Predict the model results by parsing in data**
  - Using the .predict() method, you can send data to the model deployment endpoint and it will return the predictions.

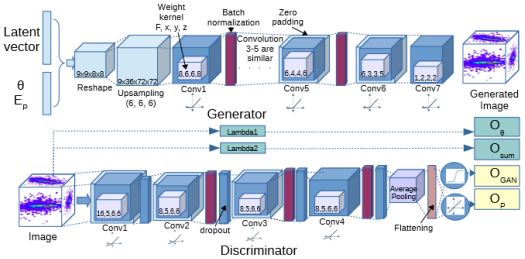
Tests realized in the Model Catalog

Two models:

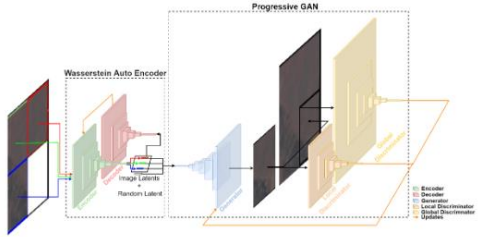
- 3DGAN for High Granularity Calorimeter images [arXiv:2109.07388]
  - Smaller model (few millions parameters)
  - Takes around 4 day to fully train with 1 GPU
- VAE-ProGAN for satellite image generation [arXiv:2211.15303]
  - Bigger model (around 80 million parameters)
  - Composed of a VAE and a Progressive GAN trained separately
  - Needs close to 32 GB of GPU memory for the whole training
  - Takes more than a month to fully train with 1 GPU
- Both models are in tensorflow 2
- Both can be parallel trained

Test:

- Training of the 3DGAN and saving to the model Catalog
- Training of the VAE part and save it to the model Catalog
- Need to use Generic model to save (unable to use TensorflowModel)
- Possible to use inference on either of the models straight from the model catalog
- **Current test:** Use of A100 for the full Progressive GAN



3dGAN Architecture



Conditional ProGAN architecture

Other tests in ADS

- Notebooks instances
  - ADS uses different conda environment tuned for each one of the applications
  - Possible to provision one instance with GPUs
- Jobs
  - Used for background training
  - Should be used for long training instead of the notebooks
- Training Pipeline
  - Multiple Jobs either in parallel or sequentially
  - Possible to make dependencies between jobs
- Distributed Training

- **Current test:** Multiple distributed options to see the impact on the training and on FLOPs

Model Catalog at CERN

Usage of the model registry to save resources

- Optimize once and everyone can use it
  - Save best weights
- Possible to save different versions
- There is no need to retrain if it is only used for inference

Beyond the ML R&D use case:

- How can we reuse a pretrained model on different tasks
  - Limited by the nature of the input and the model generalization capabilities
  - In HEP, it is usually unfeasible, inefficient or too costly in terms of data-preprocessing
- Leverage current studies on DL generalisation
  - Foundation Models

Conclusion and Future Work

Conclusion:

- Usage of a model registry as part of the development pipeline
  - Faster development
  - Easy to share work (knowledge transfer)
- Usage of the model registry to save resources
- Use the model catalog beyond the R&D use case

Future Work:

- Understand how to integrate the Model Catalog at CERN
  - Hybrid CERN-Oracle Model Catalog framework

Poster at CHEP 2023: Deploying a machine learning model catalog at CERN


CERN
openlab